Contract Deliverable for Task 23

Architecture Documentation for the Integrated Database for Mental Health and Substance Abuse Treatment Service Project

Under CSAT/CMHS
Contract # 270-96-007
Project Officer: Jon Gold

July 14, 1997

TABLE OF CONTENTS

Executive Summary... 1
Introduction 2
Architectural overview 3
  Data Processing Framework 3
  System Requirements... 8
    Hardware Platform 9
    Operating System 9
    Data Base Management System...10
    Networking and telecommunications...11
    Analysis/Reporting 11
    Client Side Software 11
Feedback from the States 12
Appendix A - Glossary...15
Appendix B - SAS and SQL..17

LIST OF FIGURES

Figure 1 - Integrating Medicaid and MH/SA Data 4
Figure 2 - Overview of Export Programs for the Final Data Sets 7
Figure 3 - Relationships Among the Data Sets in the Final Analysis.................... 8
Figure 4 - Summary of Tools Used by the States 14
Figure 5 - SQL and SAS equivalents...17

Executive Summary

This document describes the technical architecture for the Integrated Database for Mental Health and Substance Abuse Treatment Service project. In the previous contract deliverable, we identified SAS and open Relational Database Management Systems (RDBMSs) as candidate architectures. Given the transformation- and computation-intense tasks required by this project, SAS was selected as the most appropriate DBMS. In order to facilitate potential use in a more traditional RDBMS such as Oracle, the resulting database will be stored in a set of three data sets, which can be delivered as SAS data sets, SAS transport files, or ASCII text output files. This allows the resulting data to be analyzed in virtually any application on any platform.

Data sources for this project will include mental health databases, substance abuse databases and Medicaid data from three states. During data processing, records from each of the data sources will be linked using key data elements and a unique client identifier will be created. Data process involves multiple steps and subroutines in order to create the final data set.

Given the scope of the project and the large number of records involved, especially in the intermediate processing steps, we have recommended a fast (≈ 500 MHz) RISC-based computer using an advanced version of the UNIX operating system. SAS will be used as the database management system (DBMS) as well as for analysis of data quality.

During separate teleconferences with the three states, representatives from each of the states voiced strong support for the dual architecture proposed in the last deliverable. Each of the three states voiced interest in obtaining the methodological results (both good and bad) of the record linking routines, as well as receiving the entire database for comparative purposes. Other concerns involved whether there would be a mechanism for linking records from the final data sets to existing state records. Approaches to this will be addressed in a future deliverable.

Introduction

Broadly stated, the goal of this project is to develop a mental health and/or substance abuse (MH/SA) database that integrates data from multiple sources (e.g., mental health, substance abuse, and Medicaid) obtained from three states: Washington, Delaware, and Oklahoma. Once completed, the database will make possible analysis of costs and utilization for the various programs and for national estimates as well. This database will be available for on-line access by SAMHSA/CSAT/CMHS and participating state staff. Accordingly, this requires a plan for the taking in, transformation, and ultimately the structuring of the data that will comprise the database.

Previous contract deliverables have detailed the data available from each state (Task 21) and alternative structures for processing these data (Task 22). The deliverable for Task 22 described two competing architectures: one that utilized an open RDBMS architecture whereas another candidate architecture used the Statistical Analysis System (SAS). Each of the two candidate architectures was evaluated in light of two challenges, which arose early on in the analysis phase. More than any other single functional specification, these challenges would impact the design of the logical data model and system architecture. Briefly stated, these are:

- Creating a logical structure that retains the richness of each state's data while retaining a "common denominator" format to make analysis possible and:

- Developing matching routines and algorithms to link Medicaid records (which contain a Medicaid identification field) to non-Medicaid records without a common identifier.

With respect to the two competing system architectures described in Task 22, either of the competing architectures would, with varying levels of effort and add-on tools, suffice for the purposes of the current project. However, SAS was judged superior in terms of data manipulation and transformation capabilities, which are critical to the success of this project. The complexity of the matching process influenced the choice of architecture more than any single factor.

Moreover, the major perceived drawback of SAS was the relatively small installed user base compared to traditional RDBMS vendors such as Oracle, Informix, Sybase and Microsoft. In our discussions with the relevant state agencies, we found that although SAS was not used by *all* of the organizations involved, SAS was used in some capacity by each of the three states involved, and each of the states have plans to continue or expand their use of SAS. We will elaborate on this point in the Feedback from the States section on page 12.

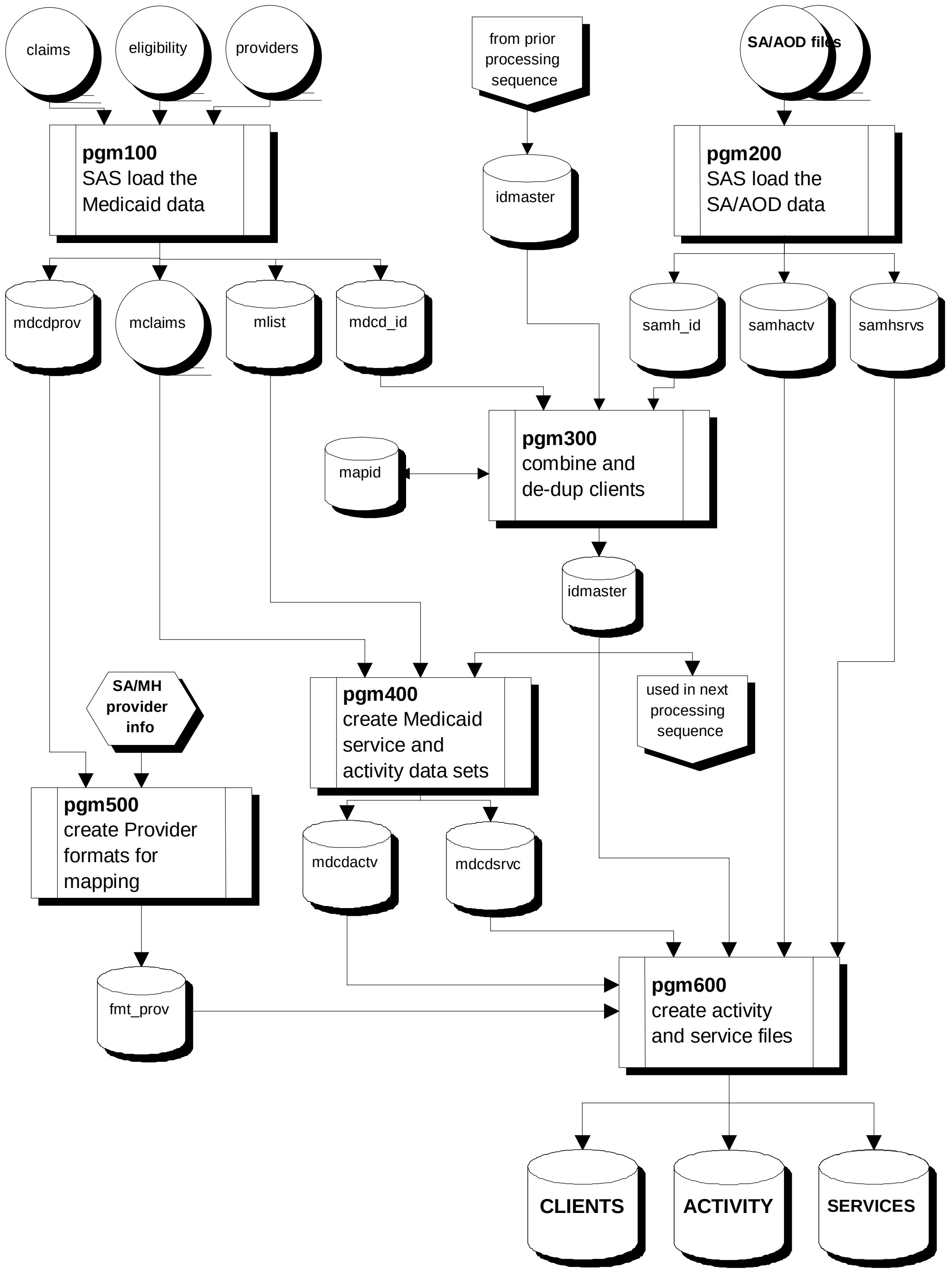
In order to accommodate those potential users who: (a) did not have access to SAS or (b) would prefer to analyze data in a traditional RDBMS, The MEDSTAT Group (MEDSTAT) has proposed using a dual architecture whereby SAS is used for input, transformation, storage and analysis and the resulting database would reflect a more “normalized” design than many typical SAS data sets. The final database would be further normalized in a series of “flat” text files that could be imported into almost any other RDBMS. This requires marginally more consideration during the database design phase, with the benefit that SAS and RDMS users can access the resulting database with almost equal ease.

Architectural Overview

Generally speaking, the database architecture is an integrated set of standards for planning, building, using, and maintaining the database. In the context of the current project, the essential components of the architecture involve the data processing framework and system requirements.

Data Processing Framework

In order to create the integrated database, data must be combined from a variety of sources. Although there are a variety of formats possible, most incoming data will be stored either as ASCII or EBCDIC files. Loading the data into the DBMS is accomplished through a series of programs, described below. A graphical representation of this process is provided in Figure 1 on page 4.



claims
eligibility
providers
from prior processing sequence
SA/AOD files
pgm100 SAS load the Medicaid data
idmaster
pgm200 SAS load the SA/AOD data
mdcdprov
mclaims
mlist
mdcd_id
samh_id
samhactv
samhsrvs
mapid
pgm300 combine and de-dup clients
idmaster
SA/MH provider info
pgm400 create Medicaid service and activity data sets
used in next processing sequence
pgm500 create Provider formats for mapping
mdcdactv
mdcdsrvc
fmt_prov
pgm600 create activity and service files
CLIENTS
ACTIVITY
SERVICES

Figure 1 - Integrating Medicaid and MH/SA Data

Program 100 is the first program in the processing sequence and loads Medicaid data from the state into SAS data sets and flags eligibles who have received MH/SA services. The program is divided into three steps to:

- Load the claims file,
- Load the eligibility file, and
- Load the provider file.

While loading and processing the Medicaid claims, the program uses special files of procedure and diagnosis codes to identify MH/SA services. Every claim record is retained in the output SAS claims data. In addition, a list of all recipients with any MH/SA service, based upon the previously mentioned files, is also output. The claims data is sorted by the Medicaid ID and service date for output. The list of MH/SA recipients is sorted by Medicaid ID where duplicate IDs are removed. During the next step in this process SAS loads the Medicaid eligibility file, combining records for eligibles with multiple eligibility periods. In the process of combining the records, an array of eligibility dates is created. Output is sorted by Medicaid ID. The last step loads the Medicaid provider file into a SAS data set, which is sorted by Medicaid provider ID.

Program 200 loads MH/SA data into SAS data sets. Because the data from each participating state is uniquely structured, it is necessary to create separate PGM200 programs for each state's data. In general, the program is designed to read the raw data files, and then create three SAS data sets: service level, summarized activity, and ID/demographics. The ID data set contains one observation for each state ID found on the incoming files. All MH/SA services are summarized in the activity data set. The service level, where available, provides information on each MH/SA service. All service level data provided by the states is retained. Service level data is not always available because of data limitations for some participating states' agencies.

The third program in the sequence, Program 300, integrates the MH/SA and Medicaid ID data, de-duplicates clients, and assigns client IDs. The Medicaid ID data from program 100 is de-duplicated, with the DEDUP include routine described below, and then matched with MH/SA ID data from program 200. Data is matched three separate times on three different variables:

- Medicaid ID,
- Social Security Number (SSN), and
- Client gender and date of birth.

After each matching step, the individual matches are scored as to the appropriateness of the match, based upon IDs, first and last names, gender, date of birth, and client ZIP code. Only matches with scores over a predetermined threshold are retained in order to minimize the number of false matches. Results from the three matches are combined and the highest score for each state ID is kept and used as the mapping for that particular ID. As a final step, state IDs not matching to any Medicaid ID are de-duplicated with the DEDUP include routine.

DEDUP is an include file, a section of code that is “included” within another program. The code for DEDUP is generic enough to enable its use in several situations. DEDUP’s purpose is to identify and link IDs in a data set that represents one client. Functionally, DEDUP is similar to program 300 except DEDUP operates on a single data set whereas program 300 operates on two data sets. Client identifiers are assigned in DEDUP.

Program 400 creates activity and service level Medicaid data. The program combines Medicaid IDs linked to state MH/SA clients in program 300 with Medicaid eligibles identified as receiving MH/SA services in program 100. All Medicaid claims for these clients are collected from the program 100 Medicaid claims data set. Medicaid service and activity level files are output.

Program 500 uses Medicaid provider data and state MH/SA provider information to create SAS formats used for provider mapping or crosswalks. SAS formats are created to map from Medicaid provider ID to provider type, from MH/SA provider ID to provider type, and from Medicaid provider to MH/SA provider ID.

The final SAS analysis data sets: (a) CLIENTS, (b) ACTIVITY, and (c) SERVICES are created by program 500. Medicaid and MH/SA activity and services data, from programs 200 and 400, are combined and duplicated services removed. The master ID data set from program 300 and the ACTIVITY data set are used to produce the CLIENTS data set containing demographic data on all clients in the final analysis data.

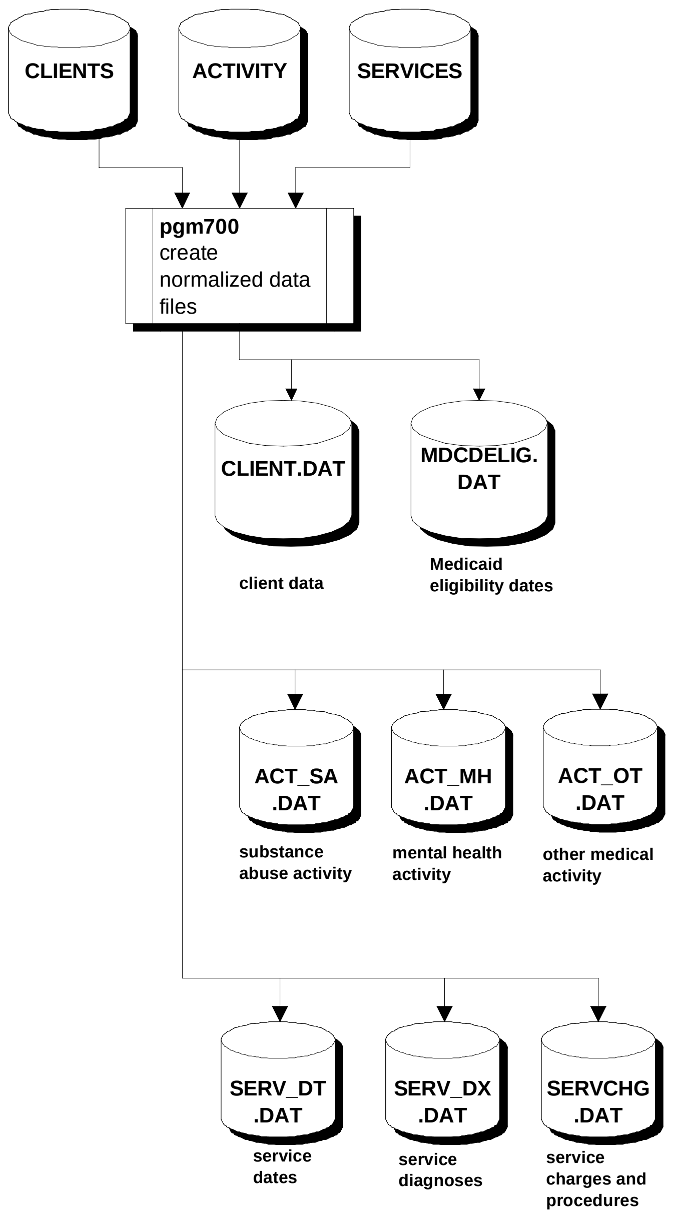


Figure 2 - Overview of Export Programs for the Final Data Sets

As noted in the ER diagram on page 8, there are two relationships between these tables. The first is a one-to-many relationship between records in the clients' data set and records in the activity data set. Each client can have one or more records in the activity table, which represent summarized charges and costs of services. A separate one-to-many relationship exists between clients and services, whereby each record in the client table has one or more associated records in the services table. The services data set maintains information regarding the diagnosis and/or procedure codes where appropriate.

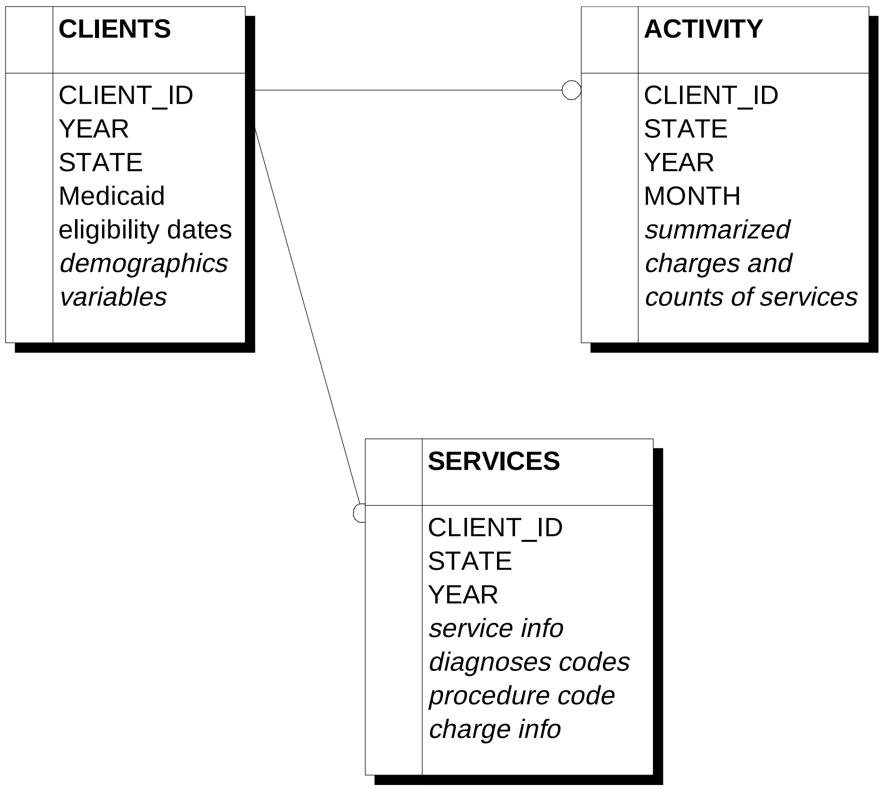


Figure 3 - Relationships Among the Data Sets in the Final Analysis

System Requirements

The size of the data, both input and output, and the complexity of the processing required to link the various data files and build the final databases, have lead us to identify a midrange UNIX server with 100 GB of disk storage as the necessary hardware for this project. Data used will vary from relatively small files with tens of thousands of records, such as MH/SA program enrollment, to very large Medicaid service level files with tens of millions of records. The largest files will exceed the 2.1 GB limit of 32-bit operating systems. Storage requirements for intermediary and final data sets will be very large, approximately 20 GB per state. Disk space of 100 GB will allow us to handle the three states and have the workspace necessary to process the data. The computational needs of reading and sorting these large files indicate the need for the processing speed and power of a midrange server. Program 300, the process of combining and de-duplicating will produce Cartesian products data sets with over 100 million observations. From our benchmarking tests, we estimate that program 300 alone would require over 24 hours to complete on a Pentium class PC.

Hardware Platform

Our benchmark tests point to the Digital Equipment Corporation (DEC) AlphaServer 4000 to provide the best performance in our price range. The DEC 4000 uses a very fast 466 MHz, 64-bit RISC CPU. Our SAS Benchmark tests indicate that this system is approximately nine times faster than a Pentium class PC. System highlights include:

- 466 MHz, 64-bit CPU with 4 MB cache
- 1 GB of RAM
- 103.2 GB disk capacity
- Seven-cartridge tape system with 140 GB storage capacity
- 10 slot, IBM 3480/3490 compatible tape drive
- 64-bit memory addressing allowing native access to files over 2.1 GB in size

Operating System

The operating system for the platform is DEC UNIX. The operating system is the low-level software that schedules tasks, allocates storage, handles the interface to peripheral hardware and presents a default interface to the user when no application program is running. UNIX was originally created circa 1970 by Ken Thompson at AT&T so he could play computer games. Since then, UNIX has become a stable, secure, and open system that is the proven standard for technical and commercial applications. UNIX systems offer a uniquely flexible and developer-friendly environment that is also very secure.

Digital UNIX is native 64-bit, standard-compliant operating system supporting nearly all-common UNIX application interfaces. A modern operating in every respect, Digital UNIX is the first 64-bit UNIX to obtain the UNIX '95 brand certification - the industry standard. Built around the power and speed of the DEC Alpha architecture, Digital UNIX provides a highly reliable and flexible, operating system. Features include:

- Advanced virtual memory - supporting files as large as 17 TB (terabytes).
- Sophisticated security capabilities - increased protection of sensitive data.
- Memory mapped file support - freeing applications from managing file I/O, increasing throughput and decreasing overhead.

Basic security is built into Digital UNIX and meets all C2-level features. Most UNIX systems are lower level C1. Features include:

- Event and application auditing,
- Audit analysis,
- User identification and authentication,
- Login controls and password aging,
- Access control lists, and
- Discretionary access controls.

Digital UNIX also adds B1/CMW level security tools and features such as trusted windowing and networking.

Database Management System

SAS software is a combination of a statistical package, a database management system, and a high level programming language. Produced by the SAS Institute in Cary, North Carolina, it serves as an excellent tool for data management and statistical analyses. SAS originated in the academic and research communities and now serves as an all-purpose corporate information system, running on a wide range of operating systems. The basic unit of data storage in SAS is a data set, which is structured like a rectangular file - observations are rows and variables are columns. Data sets contain both data values and descriptor information which describe the contents. There is virtually no limit to the number of observations and variables in a data set.

The SAS programming language is very useful and extremely versatile. Employing traditional programming techniques with a wealth of built-in functions and pre-compiled procedures, experienced SAS programmers can quickly develop powerful and useful programs. While SAS is an interpreted language, it is still extremely fast. Performance tests on an IBM mainframe have shown SAS to outperform faster than pre-compiled COBOL programs on many tasks. The data manipulation and transformation capabilities of SAS are outstanding and those capabilities will greatly benefit this project. The matching process and other steps in this project will require sophisticated and complex data processing and programming and the SAS programming language offers excellent facilities for these tasks. MEDSTAT has a staff of highly skilled SAS programmers who can take full advantage of the SAS programming language.

Networking and telecommunications

The platform will link to MEDSTAT's Santa Barbara Novell LAN via a TCP/IP connection. This link will enable connections to printers and programmer's workstations. For security reasons and due to the sensitive nature of the data, there will be no direct connections to the server outside of the Santa Barbara office - either through telephone lines or connections to the Internet. Nor will such a connection be necessary since all programmers associated with the project operate from MEDSTAT's Santa Barbara office.

Analysis/Reporting

Although the scope of the project does not call for any analysis activity, there is a need for data analysis and reporting during processing to ensure data quality. Processing work and data will be in SAS, which is a full-featured analysis and presentation application. SAS can perform a variety of data analysis and presentation functions, including statistical analyses and graphical presentation of data. Quality analysis will be performed as the data is processed, saving both time and effort.

Client Side Software

The ultimate goal of this project is to create an integrated database (for analytic use) of all medical activity for persons receiving MH/SA treatments. SAMHSA and at least one agency in each of the three states use SAS in some way, and the final databases will be in SAS format. For these users, database delivery will be in the form of SAS data sets in transport format, which are a machine independent file format used for exchanging SAS data sets between platforms (e.g., between a PC running Windows 95 and a mid-range computer running UNIX). Agencies who wish to use the databases in SAS format will only need a current copy of the SAS system to use the data.

Users will also have the option of receiving data in a format that can be imported into SQL Server, FoxPro, or any other RDBMS. A series of normalized, "flat" text files will be created from the final SAS data sets. To use the data in a RDBMS such as SQL Server or FoxPro users will need to load the data files with a bulk copy utility.

The requirements for client side software are, by design, as open as possible. Although we anticipate that SAS running under Windows NT on an Intel Pentium Pro class machine will be sufficient for most analyses, the data sets can be delivered as SAS transport files, which can be imported in SAS in any environment on any platform. Additionally, SAS transport files can be imported into other common statistical analysis packages such as SPSS.

Although most traditional RDBMS packages do not read SAS or SAS transport files, the data sets will also be made available as ASCII text files which can be imported into almost any RDBMS application. Some database systems have inherent limitations as to the number of columns and or rows. Although this is typically not a problem with higher-end systems (e.g., those from Oracle, Sybase, Informix, and Microsoft SQL Server), many desktop applications manifest these limitations. For example, Microsoft Access tables are limited to no more than 255 columns (fields) and no more than 2 billion rows.

Feedback from the States

As part of the technical review process, MEDSTAT held separate teleconferences with representatives of each state's (a) MH programs, (b) SA programs, (c) Medicaid programs, and (d) other relevant agencies as appropriate. These teleconferences were held within two weeks after the Task 22 deliverable was submitted to the GPO and all of the participants had reviewed the document. Each of the teleconferences lasted between 30 and 90 minutes, and all participants had an opportunity to ask questions or clarification on any point.

Overall, there was broad-based support for the dual architecture approach advanced by MEDSTAT, and considerable support for SAS both as a data manipulation environment and as a file format for distribution of the database. At least one organization in each state indicated that they could read SAS files, either directly into SAS or through an existing application (e.g., SPSS).

Three common themes emerged during these discussions. All of the states expressed an interest in whether MEDSTAT would:

- Make the results of the matching process available
- Provide the entire database to the states
- Include a mechanism for relating records in the resulting database back to existing state data

With respect to the first point, the states were unanimous in their request for information as to what worked in the matching routines as well as what did not work. Each of the states commented that one of the most useful outcomes of this project was information on the relative success rates for various methods of matching records without a common identifier. Many of the state organizations would like to implement this type of linkage but are reluctant given the considerable investment of resources in trying to find the "best" matching routine. MEDSTAT could provide the states with information about what fields (and combination of fields) offered the highest degree of matching success in creating the integrated database. Perhaps just as useful though, is what information is not useful in matching records without a common identifier. Each state commented that knowing which combination of data elements were "dead ends" in matching process would inform and direct their own matching efforts.

States were also interested in obtaining the entire database for benchmarking and performance comparison purposes. Each of the states viewed their involvement in this project as a unique opportunity to obtain comparative data from similar agencies and programs, and were enthusiastic about the opportunity to perform internal program evaluation and analysis using the combined data from this project.

Related to the first point, many state agencies have other databases that contain information related to that in the current project. A common question from the states related to the creation of the unique client identifier, and whether that identifier would be decipherable by the states. Although outside the scope of this task, there are at least three methods for delivering the client identifier, two of which would allow the states to link the unique client identifier to their own records. These methods will be described further in a future deliverable.

	Delaware	Oklahoma	Washington
SAS	X	X	X
Oracle			X
Microsoft SQL Server			X
SPSS	X		
FoxPro		X	

Figure 4 - Summary of Tools Used by the States

Appendix A - Glossary

C2 Security - A standard from the US Government National Computer Security Council (an arm of the U.S. National Security Agency), "Trusted Computer System Evaluation Criteria, DOD standard 5200.28-STD, December 1985" which defines criteria for trusted computer products.

Denormalize – To allow redundancy in a table to that a database table can remain flat, rather than normalized. Data is often denormalized for statistical analysis and OLAP.

Equi-join – A type of join where table rows are combined only if the value of a column in the first table is equal to the value of the column in the second table. In a SQL expression, a WHERE clause or and ON clause contain the column names that must satisfy the requirement.

ER diagrams (**E**ntity-**R**elation diagrams) – A type of diagram used in data modeling for relational databases.

Join – Combining data from two or more tables, resulting in a single, new table or view. A general join combines each row from one table with all rows of the other tables, forming the Cartesian product of the original tables.

Logical design – The phase of a database design concerned with identifying the relationships among the data elements.

Normalize – The process of removing redundancy by separating data into multiple tables. One of the constraints imposed on a relational database.

OLAP (**O**n-**L**ine **A**nalytic **P**rocessing) – A loosely defined set of principles that provide a framework for decision support. Generally using a multidimensional database model with appropriate access and analysis tools, OLAP primarily involves aggregating large amounts of diverse data. OLAP can involve millions of data items with complex relationships. Its objective is to analyze these relationships, look for patterns, trends, exceptions, and turn them into meaningful information. An OLAP system should support logical and statistical processing of results without the end user having to submit the request in language like SQL or C++.

OLTP (**O**n-**L**ine **T**ransaction **P**rocessing) – The process of capturing transactions, and the data relevant to them, in real time. An OLTP is primarily concerned with adding, updating, inserting and deleting records. Most frequently used with reference to relational databases.

One-to-many relationship – A logical data relationship in which the value of one data element in a given table can exist in combination with many values of a data element in another table, but not vice versa.

Physical design – The phase of database design following the logical design that identified the actual database tables and index structures used to implement the logical design.

Query – A request for information, generally as a formal SQL command passed from the front-end application to a database or search engine.

RDBMS (**R**elational **D**ata **B**ase **M**anagement **S**ystem) – A database that organizes and accesses data according to relationships between data items. Based on the relational model developed by E.F. Codd, RDBMS define data structures, storage and retrieval operations, and integrity constraints. The data and relations are organized in tables, which are collections of records. Each record in a table contains the same fields. Records in different tables may be linked if they have the same value in one particular field in each table.

Reflexive join – Joining a single table with itself.

RISC (**R**educed **I**nstruction **S**et **C**omputer) – A processor whose design is based on the rapid execution of a sequence of simple instructions rather than on the provision of a large variety of complex instructions (as in a Complex Instruction Set Computer).

SQL (**S**tructured **Q**uery **L**anguage) – A language which provides a user interface to an RDBMS. SQL is the de facto standard for RDBMS and can be embedded in other programming languages. SQL also provides basic functions for defining and manipulating tables of data.

Appendix B: SAS and SQL

Although SAS and SQL were developed independently for different purposes, there are some commonalties between the two languages. Tables in a traditional RDBMS are roughly equivalent to a SAS data set, and a SAS merge statement is similar to JOIN statement in SQL. The table below summarizes some of the common functions.

SQL/RDBMS term	SAS term
Table	Data set
Row	Observation
Field or Column	Variable
Join	Merge

Figure 5 - SQL and SAS equivalents